Appendix
乱数マスクを用いて差分電力解析に対抗

佐藤 証，高橋芳夫

DPA（差分電力解析）対策にはさまざまな方式が提案されていますが，ここではもっとも基本的なAkkarらによる乱数マスクを用いる手法を紹介します．この手法は，中間データに対して乱数をXOR（排他的論理和）することで，攻撃者が中間データを正しく予測できないようにするものです．（筆者）

Akkarの対策を用いた場合，正しい結果を得るためには，XORした乱数による影響を最終的にキャンセルする必要があります．ビット置換（DESにおけるIP，IP^{-1}，E，P）のような線形変換だけであれば，XORした乱数のビットを並べ替えてもう一度XORすることでキャンセルできます．しかし，非線形変換であるS-Boxはそう簡単にはいきません．

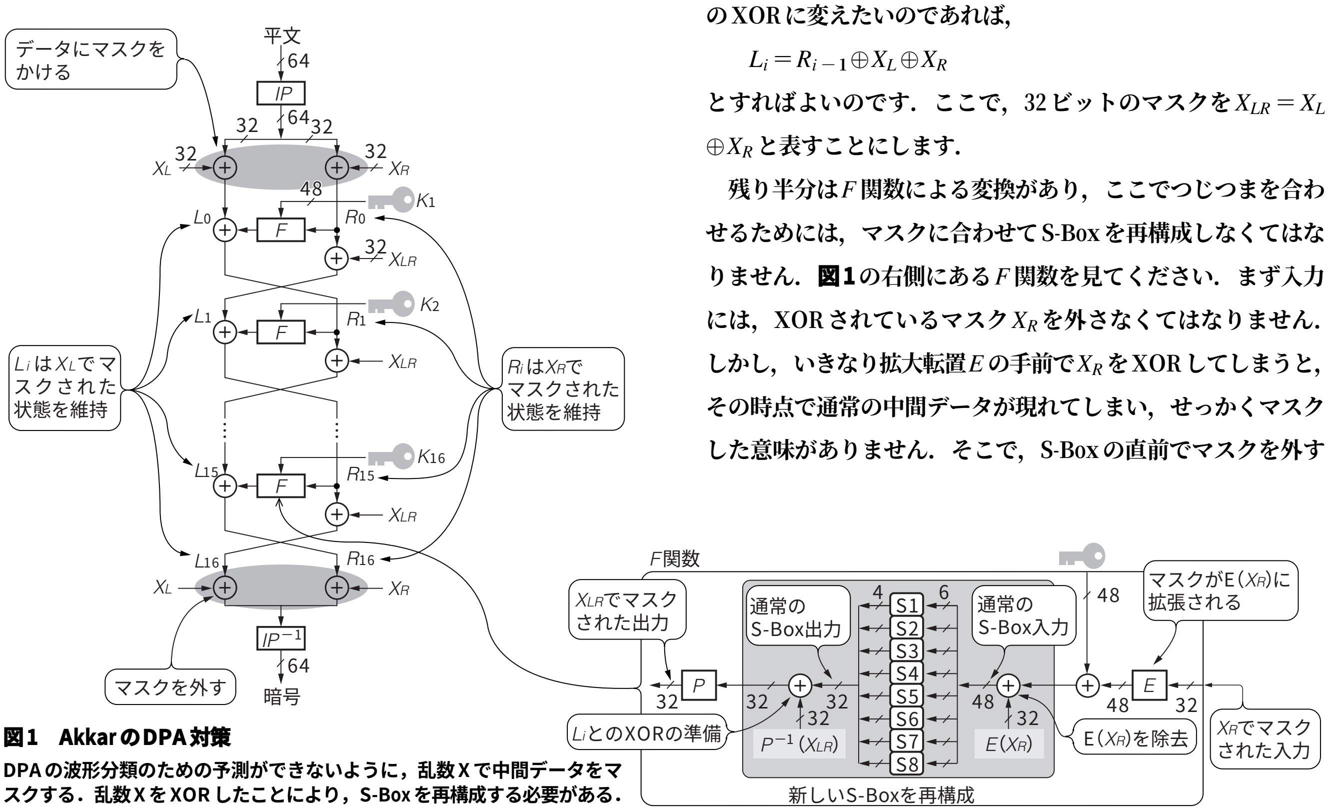


図1　AkkarのDPA対策
DPAの波形分類のための予測ができないように，乱数Xで中間データをマスクする．乱数XをXORしたことにより，S-Boxを再構成する必要がある．

● S-Boxの中間データを見破られないようにするしくみ

そこで，入力と出力に対する乱数マスクで演算がおかしくならないように，S-Boxを再構成します．これを，**図1**を用いて説明しましょう．

平文データをマスクするために，まず乱数X（左半分をX_L，右半分をX_Rとする）を生成してXORします．もともとの手法では初期転置IPの前で行いますが，効果は同じなので，話を簡単にするため，ここではIPの後で行っています．また，暗号文出力直前のデータに対して最終（逆）転置IP^{-1}の手前でXをXORすることでマスクを外します．したがって，ラウンド処理の過程において中間データL_iはXの左半分X_Lで，R_iは右半分X_Rでマスクされた状態を維持できれば，正しく暗号化できます．

通常のDES処理において$R_{i-1}\rightarrow L_i$の変換は配線をねじるだけなので，R_{i-1}にX_RがXORされていて，L_iではそれをX_LとのXORに変えたいのであれば，

$$L_i = R_{i-1} \oplus X_L \oplus X_R$$

とすればよいのです．ここで，32ビットのマスクを$X_{LR} = X_L \oplus X_R$と表すことにします．

残り半分はF関数による変換があり，ここでつじつまを合わせるためには，マスクに合わせてS-Boxを再構成しなくてはなりません．**図1**の右側にあるF関数を見てください．まず入力には，XORされているマスクX_Rを外さなくてはなりません．しかし，いきなり拡大転置Eの手前でX_RをXORしてしまうと，その時点で通常の中間データが現れてしまい，せっかくマスクした意味がありません．そこで，S-Boxの直前でマスクを外す

表1　スクランブル追加版へのDPA結果

S1	S2	S3	S4	S5	S6	S7	S8
32	35	52	3	46	7	0	50
50	35	52	3	21	7	0	50
20	35	52	56	46	24	38	50
50	35	37	3	46	7	1	50

(**a**) サンプル数：1,000　エラー：8

S1	S2	S3	S4	S5	S6	S7	S8
23	35	52	3	46	7	0	50
50	35	52	3	46	7	0	46
50	35	52	48	46	57	0	50
50	35	52	3	46	7	0	50

(**b**) サンプル数：3,000　エラー：4

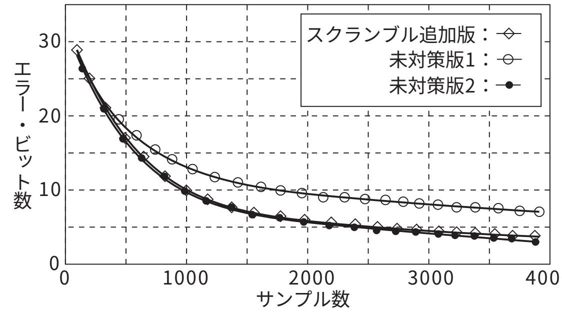


図2
スクランブルの有無とエラー・レートの関係
未対策版1は第3章の図13と同じデータ．未対策版2は，1とほぼ同じコードだが，配置配線が異なるもの．スクランブルの有無よりも，配置配線などによる回路の違いのほうがDPAへの影響が大きいことがわかる．

ことを考えます．

データはEによって48ビットに拡大されているので，マスクも32ビットのX_Rではなく，$E(X_R)$と48ビットに拡大したものをXORします．これでS-Boxから本来の正しい出力が得られます．そして，32ビット転置Pを通してL_{i-1}とXORの後にR_iとするのですが，L_{i-1}はX_Lでマスクされているので，これをX_Rに変えなければなりません．これもF関数の出力にX_{LR}をXORするのではなく，転置Pの手前で行うことにします．これら一連の変換を式でまとめると，次のようになります．

$$
\begin{aligned}
R_i &= L_{i-1} \oplus F(R_{i-1}) \\
&= L_{i-1} \oplus P(SBox(E(R_{i-1} \oplus X_R) \oplus K_i)) \oplus X_{LR} \\
&= L_{i-1} \oplus P(SBox(E(R_{i-1} \oplus K_i \oplus E(X_R)) \oplus P^{-1}(X_{LR}))
\end{aligned}
$$

ところでこの乱数マスクを施したF関数において，S-Boxへの入出力はマスクを外した本来の値であると説明しました．これでは，この中間データをDPAで予測されてしまうおそれがあります．しかし，実際には**図1**に示したように，$E(X_R)$と$P^{-1}(X_{LR})$を元のS-Boxと合成した新たなS-Boxをランダム・テーブルとして再構成しておきます．こうすることで，元のDESと同じ中間データが計算途中で出現することはなくなります．

乱数は，DESの処理の16ラウンドごとに生成する必要はありません．しかし，適当なサイズのブロックを暗号化するたびに生成し，S-Boxを再構成します．したがって，S-Boxはレジスタやメモリで実装することになります．また，FPGAの場合はS-Boxの部分を変えてネットリストを再ロードすることが考えられます．自己再構成アーキテクチャのチップなども利用可能かもしれませんが，オーバヘッドの大きさが問題となりそうです．

暗号回路は，スマート・カードのようにプロセッサを持つセキュリティ・システムの1コンポーネントとして組み込まれることが多いため，そのような場合はプロセッサで乱数生成とS-Boxの再構成を行えばよいでしょう．

● Akkarの対策に対してDPAを試みる

ここでは話を簡単にするため，Akkarの対策において乱数マスクをX＝0x123456789abcdefで固定としたVerilog HDLのサンプル・コードを用います（誌面のつごうで割愛するが，本サンプル・コードはhttp://www.aoki.ecei.tohoku.ac.jp/crypto/からダウンロードできる）．乱数XがわからなければDPAで選択関数の出力が‘0’か‘1’か判断できないので，固定でもかまわないと考える方もいるかもしれません．それではその予想が正しいかどうか検証してみることにしましょう．

表1は，上述のサンプル・コードを使い，本特集の第3章と同じ環境でDPAを行った結果です．また，**図2**はエラー・レートをプロットしたものです．比較のため，未対策コードの結果を二つプロットしてあります．意外なことに，中間データを乱数でマスクするしないにかかわらず，正しい秘密かぎが得られ，またエラー・レートも未対策版とほぼ同じであることがわかります．

実はDPAでは，注目したビットが‘0’か‘1’かを正しく予想する必要はなく，両者が区別できることが重要なのです．乱数が未知であっても固定されているならば，それとXORされる中間データは特定のビットがつねに反転しているに過ぎません．例えば，注目しているビットのマスクが‘1’の場合，データが‘0’であれば‘1’に，‘1’であれば‘0’になるので，両者を正しく分類できます．

この実験結果から，Akkarの対策を用いる場合は少なくとも数百ブロックごとにS-Boxを変更する必要があり，そのままではFPGAにおける対策としてはあまり実用的な方法とは言えません．ですが，乱数マスクや演算処理の変形はサイドチャネル攻撃への対策法の基本的なアイデアであり，さらに実用的な対策法も多数考案されています．

さとう・あかし　日本アイ・ビー・エム（株）東京基礎研究所
たかはし・よしお　（株）NTTデータ 技術開発本部